Algorithm-Hardware Co-optimization for State-of-the-Art Optimal Control Solvers on Embedded SoC

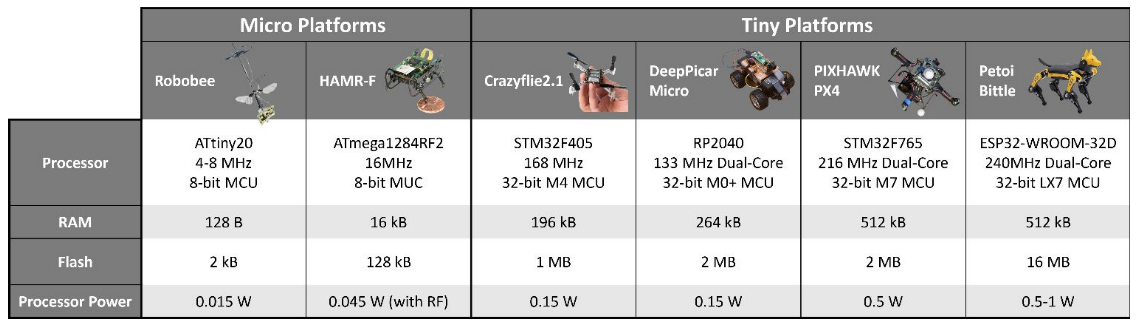
Kris Shengjun Dong, Dima Nikiforov, Minh Nguyen, Borivoje Nikolic, Sophia Shao

SLICE
Berkeley
UNIVERSITY OF CALIFORNIA

Overview

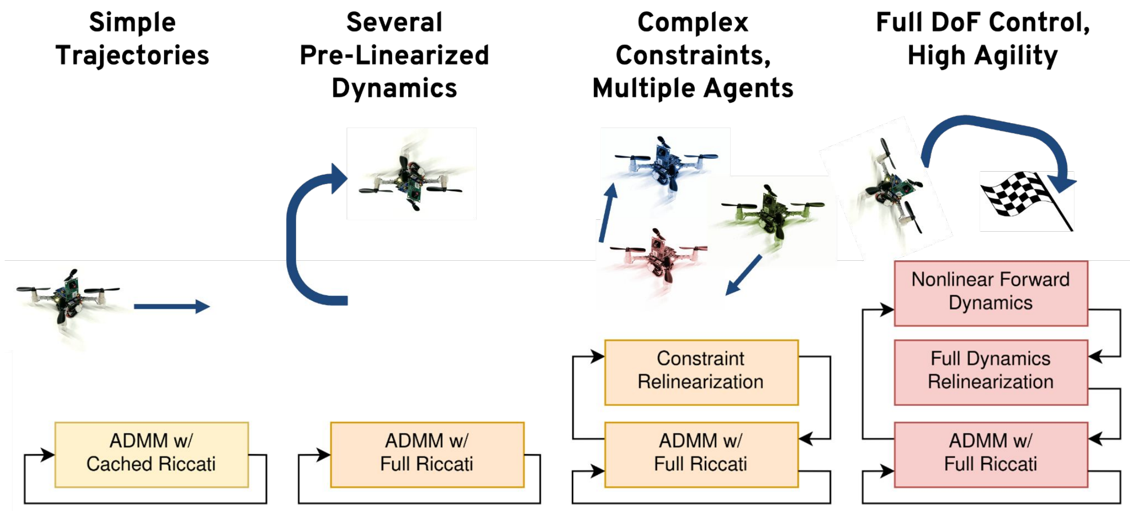
Real world robots with embedded SoCs are often computationally constrained and can not utilize onboard optimal control algorithms such as Model Predictive Control (MPC). We apply algorithm-hardware co-optimization for GEMV operations to accelerate MPC for these cases.

Range of Robotic Platforms



	Micro Platforms		Tiny Platforms			
	Robobee	HAMR-F	Crazyflie2.1	DeepPicar Micro	PIXHAWK PX4	Petoi Bittle
Processor	ATtiny20 4-8 MHz 8-bit MCU	ATmega1284RF2 16MHz 8-bit MUC	STM32F405 168 MHz 32-bit M4 MCU	RP2040 133 MHz Dual-Core 32-bit M0+ MCU	STM32F765 216 MHz Dual-Core 32-bit M7 MCU	ESP32-WROOM-32D 240MHz Dual-Core 32-bit LX7 MCU
RAM	128 B	16 kB	196 kB	264 kB	512 kB	512 kB
Flash	2 kB	128 kB	1 MB	2 MB	2 MB	16 MB
Processor Power	0.015 W	0.045 W (with RF)	0.15 W	0.15 W	0.5 W	0.5-1 W

Complexity of MPC



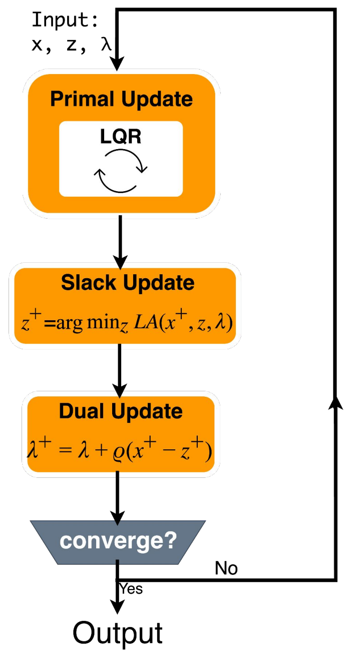
Simple Trajectories

Several Pre-Linearized Dynamics

Complex Constraints, Multiple Agents

Full DoF Control, High Agility

Target Workload



Input:
x, z, λ
Primal Update
LQR
Slack Update
z^{+}=arg min$_z$ $LA(x^{+}, z, \lambda)$
Dual Update
$\lambda^{+} = \lambda + \varrho(x^{+} - z^{+})$
converge?
No
Yes
Output

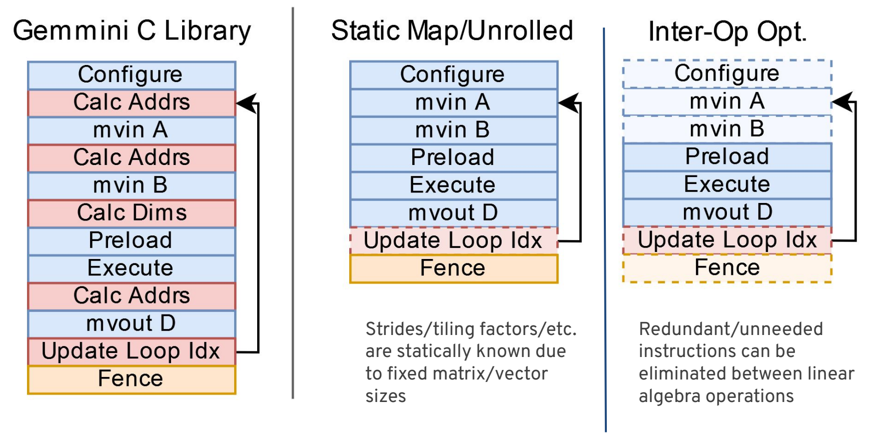
$$\text{primal update}: x^{+} = \arg\min_{x} \mathcal{L}_A(x, z, \lambda),$$

$$\text{slack update}: z^{+} = \arg\min_{z} \mathcal{L}_A(x^{+}, z, \lambda),$$

$$\text{dual update}: \lambda^{+} = \lambda + \rho(x^{+} - z^{+}),$$

Methodology

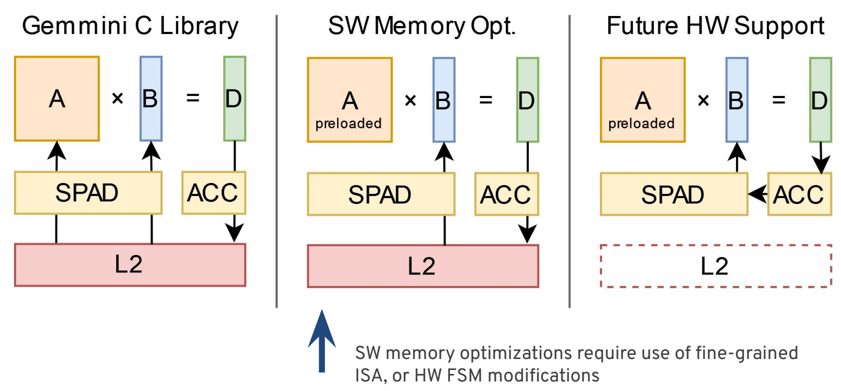
Static Mapping



Strides/tiling factors/etc. are statically known due to fixed matrix/vector sizes

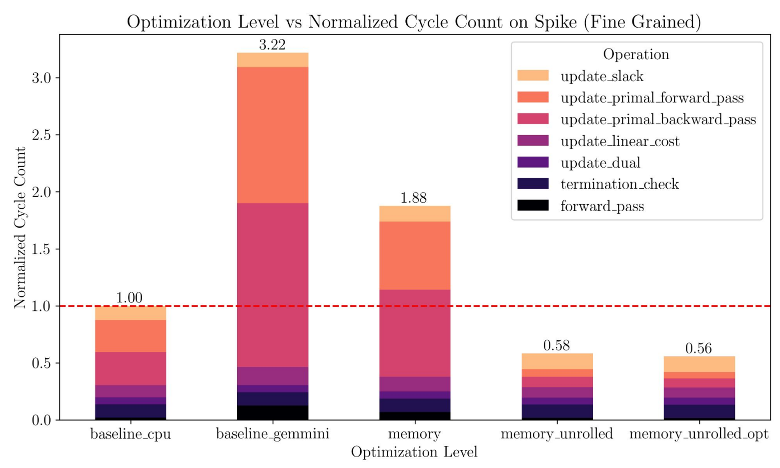
Redundant/unneeded instructions can be eliminated between linear algebra operations

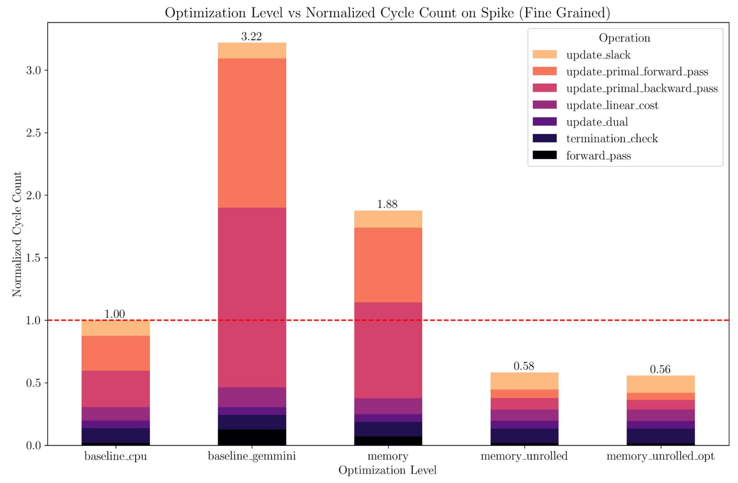
Memory Optimizations

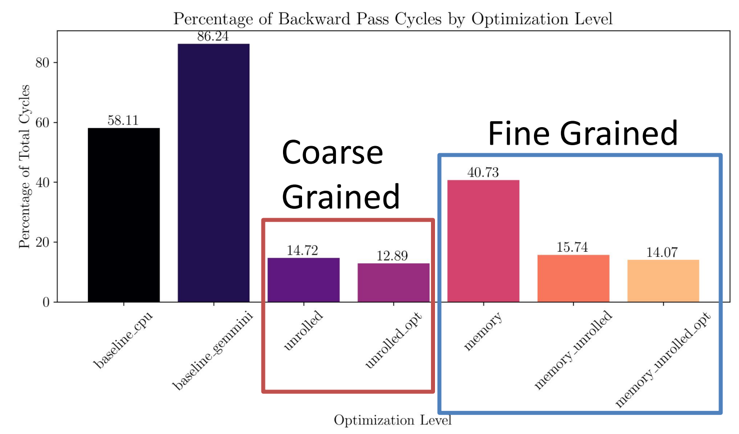


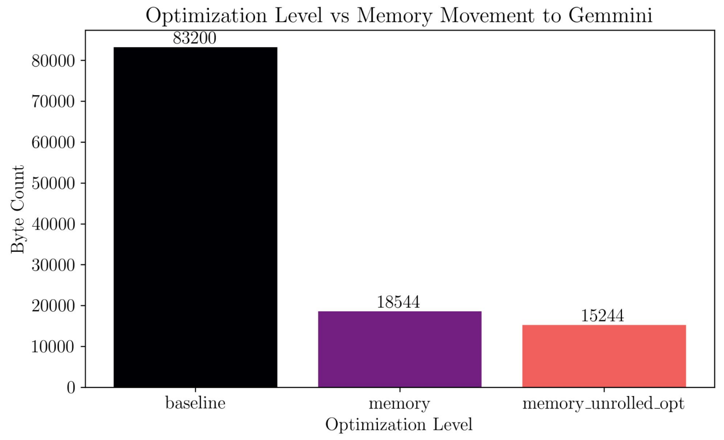
SW memory optimizations require use of fine-grained ISA, or HW FSM modifications

Spike Evaluation



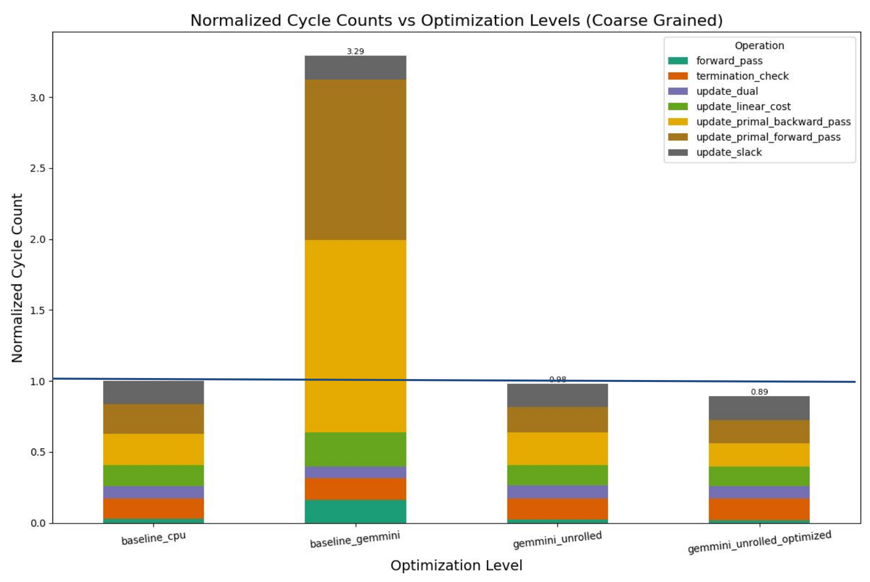




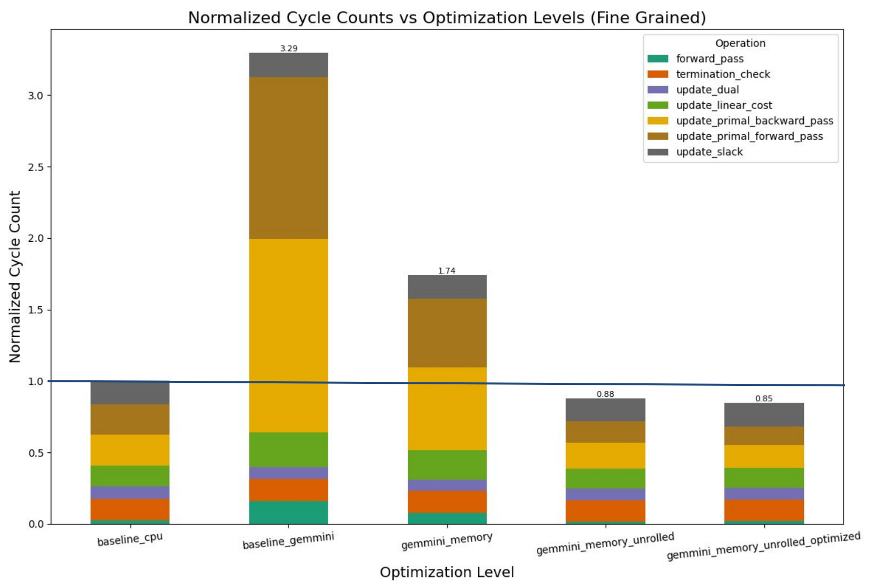


FireSim RTL Evaluation

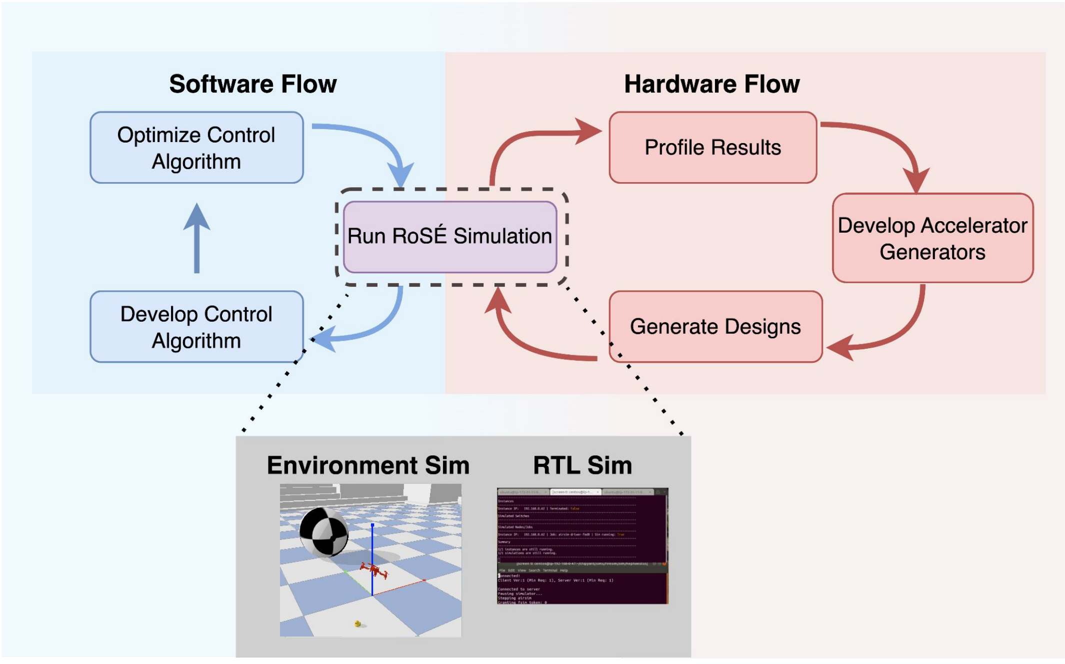
Coarse Grained Gemmini



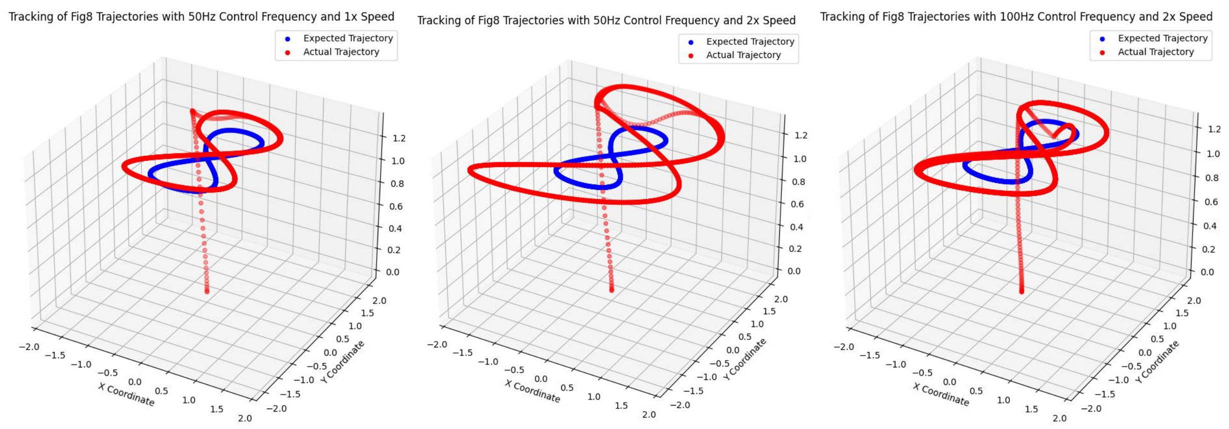
Fine Grained Gemmini



Evaluation



Performance's Impact on Quality of Flight



Control Frequency	50Hz	50Hz	100Hz
Speed	1x	2x	2x
Tracking Error	0.73453	1.48734	0.77418

Conclusion

- Initially mapping dynamic GEMV operations to Gemmini results in worse performance than Eigen CPU implementation
- Hand-tuned unrolling and optimizations outperforms CPU

Future Work

- Implementing HW features to support fine-grained GEMM/GEMV operations
- Codegen/Compiler support for end-user utilization of optimized kernels

Contact Information

Kris Shengjun Dong	krisdong@berkeley.edu
Dima Nikiforov	vnikiforov@berkeley.edu
Minh Nguyen	minh02@berkeley.edu

Jan 2024